REMARKS/ARGUMENTS

Claims 1-20, are pending in the present application. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Obviousness

I.A. Claims 1-5, 8-13 and 16-20 over *Widell et al.* in view of *Lordi et al.*

The Examiner has rejected claims 1-5, 8-13 and 16-20 under 35 U.S.C. § 103(a) as being unpatentable over *Widell et al.*, Collision Handling Apparatus and Method, Publication No. US 2005/0055490, published March 10, 2005 (hereinafter "*Widell*") in view of *Lordi et al.*, Restoring the State of a Set of Files, Patent No. 5,857,204, dated January 5, 1999 (hereinafter *"Lordi"*). This rejection is respectfully traversed.

Applicants' claim 1 recites "change a first set of data for a first thread associated with a first file of said filesystem".

Applicants' claim 8 recites "changing a first set of data for a first thread associated with a first file of said filesystem".

Applicants' claim 16 recites "first instructions for changing a first set of data for a first thread associated with a first file of a filesystem".

The Examiner asserts that *Widell* teaches "change a first set of data for a first thread associated with a first file" in paragraph 43, lines 4-6, and paragraphs 34, 36, and 39. The Examiner also asserts that *Widell* teaches that many different types of data structures may be used, which would include a file. It appears that the Examiner is equating a data structure and a file.

Widell's paragraphs 43, 34, 36, and 39 are reproduced below.

> [0043] The embodiments of the present invention described above comprise data structures in the form of bit vectors for storing information indicative the thread's accesses to the memory. However, many alternative types of data structures for storing this information are possible according to the present invention. The data structures may for instance be implemented as lists to which numbers that correspond to the memory elements are added to indicate accesses the memory elements. Other possible implementations of the data structures include trees, hash tables and other representations of sets.
>
> [0034] According to the present invention each thread 5, 6, 7 is associated with a data structure 9, 10, 11, which is illustrated schematically in FIG. 1. The data structure is used to store information indicative of which memory elements in the shared memory 4 that the respective thread has accessed. According to an embodiment of the present invention each data structure includes a number of bits 12 that correspond to the memory elements in the shared memory. According to the embodiment of the present invention shown in

FIG. 1 the bits 12 of each data structure 9, 10, 11 are divided into a load vector 9a, 10a, 11a and a store vector 9b, 10b, 11b. For each memory element m0, m1, m2, mn in the shared memory 4, there is exactly one corresponding bit 12 in the load vector and exactly one corresponding bit 12 in the store vector associated with each thread. When the thread 6 reads from a memory element, it sets the corresponding bit 12 in the load vector 9a to indicate that the memory element has been read. The store vector 9b is updated analogously when the thread 6 writes to the shared memory.

[0036] FIG. 3A illustrates an example of how the data structures 9, 10, 11 are used according to the present invention. In this example the thread 5 has written to the memory elements m1 and m4 and read memory elements m1, m5 and m8. The thread 6 has written to the memory elements m2, m6 and m9 and read the memory elements m2, m6 and m13. The thread 7 has read the memory element m12. In this example, there are more memory elements in the shared memory than there are bit positions in the load and store vectors, which means that there is a many-to-one correspondence between the memory elements and the bits in the load and store vectors. In this example the bit position in the load and store vector that corresponds to a selected memory element is found using a hash function, which in this example simply calculates the remainder when dividing the number of the memory element by the size of the load and store vectors. This means that when the thread 5 writes to the memory elements m1, it sets the bit in position number 1 in its store vector and when the thread 6 writes to the memory element m9, it sets the bit in position number 1 in its store vector. When the threads have performed the write and read operations mentioned above, the bit position numbers that are set will be 0, 1, 5 for the load vector 9a; 1, 4 for the store vector 9b; 2, 5, 6 for the load vector 10a; 1, 2, 6 for the store vector 10b and 4 for the load vector 11a. This is illustrated in FIG. 3A by means of filled boxes representing the bits that are set.

[0039] The embodiments of the present invention shown in FIGS. 3A and 3B uses a type of data versioning called privatisation, which means that a private copy 14 of a memory element that is to be modified is created for the thread that modifies the element. The thread then modifies the private copy instead of the original memory element in the shared memory. The private copies contain pointers 15 to their corresponding original memory element in the shared memory. The private copies are used to write over the original memory elements in the shared memory 4 when the threads for which they were created are committed. If a thread is rolled back, its associated private copies 14 are discarded. FIG. 4 shows a flow diagram illustrating how reading from the shared memory is performed when privatisation is used. FIG. 5 shows a corresponding flow diagram for writing to the shared memory.

Applicants respectfully disagree that *Widell* teaches "change a first set of data for a first thread associated with a first file". *Widell* does not teach a first set of data for a first thread associated with a first file. The paragraphs reproduced above do not describe a file. *Widell* teaches data structures. The data structures taught by *Widell* are not files, and are not analogous to files.

Widell teaches the data structure being used to store information. A thread is associated with a load vector and a store vector. Each data structure includes a number of bits. There is one bit in the load vector that corresponds to each memory element. There is one bit in the store vector that corresponds to each memory element.

The individual bits are capable of being accessed by a thread. When a thread reads from a memory element, it sets the corresponding bit in the load vector data structure. When a thread writes to a memory element, it sets the corresponding bit in the store vector data structure.

Microsoft Computer Dictionary, Fifth Edition, published 2002, defines "vector" as "In data structures, a one-dimensional array—a set of items arranged in a single column or row." An "array" is defined as "In programming, a list of data values, all of the same type, any element of which can be referenced by an expression consisting of the array name followed by an indexing expression. Arrays are part of the fundamentals of data structures, which, in turn, are a major fundamental of computer programming."

As this definition makes clear, a "data structure" is not a "file". *Widell's* use of the term "data structure" appears to be consistent with the commonly understood definition of that term. *Widell* does not describe its data structure as being a file. Therefore, *Widell* does not teach a thread that is associated with a file. *Widell* teaches a thread that is associated with an array. Because *Widell* does not teach a thread that is associated with a file, the combination of *Widell* and *Lordi* does not render Applicants' claims obvious.

The Examiner states that *Widell* teaches that many types of data structures may be used. The Examiner then states that these data structures would include a file. Applicants respectfully disagree. The data structure of *Widell* is an array, not a file.

The Examiner stated that *Widell* does not teach a filesystem having files. The Examiner relies on *Lordi* to supply the features that are missing from *Widell*. The Examiner states that *Lordi* teaches a filesystem having files in column 5, lines 1-7, and that it would have been obvious to have modified *Widell* by the teaching of *Lordi*. Applicants respectfully disagree.

Widell and *Lordi* cannot be combined as suggested by the Examiner. *Lordi* teaches files and file operations. The concepts taught by *Lordi* cannot be applied to a "data structure".

Lordi uses the term "file" in a manner that is consistent with the commonly understood definition of the term.

> In the following description, the term "path" will be used in the UNIX sense of a file name. However, adopting this usage is not in any way intended to limit the invention to UNIX embodiments. Further, by common usage, the term "path" may also be used, in a form of shorthand, to refer to the named file itself. Whether a file name or a file is indicated is determined from the context.

Lordi, column 5, lines 36-42.

Lordi clearly teaches a file. Thus, the teachings of *Lordi* apply to "files" and not "data structures" or "arrays".

Because *Widell* does not teach a first thread that is associated with a file, and because *Widell* and *Lordi* cannot be combined as suggested by the Examiner, the combination of *Widell* and *Lordi* does not render Applicants' claims obvious. Therefore, the rejection of claims 1-5, 8-13 and 16-20 under 35 U.S.C. § 103(a) has been overcome.

I.B. Claims 6, 7, 14 and 15 over *Widell* in view of *Lordi*, and further in view of *Carter*

The Examiner has rejected claims 6, 7, 14 and 15 as being unpatentable over *Widell* in view of *Lordi*, and further in view of *Carter et al.*, Remote Access and Geographically Distributed Computers in a Globally Addressable Storage Environment, Patent No. 5,987,506, filed November 16, 1999 (hereinafter "*Carter*"). This rejection is respectfully traversed.

Applicants' claims describe wherein the first file comprises an inode page, and wherein the second file comprises a directory page.

The Examiner states:

> With respect to claims 6 and 14, Widell as modified teaches claims 1 and 8. Widell as modified does not teach wherein said first file comprises an inode page.
> Carter teaches remote access and geographically distributed computers in a globally addressable storage environment (see abstract), in which he teaches wherein said first file comprises an inode page (column 30 lines 59-61).
> It would have been obvious to a person having ordinary skill in the art at the time the invention was made to have further modified Widell by the teaching of Carter because wherein said first file comprises an inode page would enable Widell's mechanisms to be implemented on a computer network system, having adaptable system configurations for dynamically exploiting distributed network resources and improved fault tolerance (Carter, column 2 lines 55-67).
> With respect to claims 7 and 15, Widell as modified teaches wherein said second file comprises a directory page (Carter, column 30 lines 60-61).

The combination of *Widell*, *Lordi*, and *Carter* does not render Applicants' claims obvious because the combination does not teach a first thread that is associated with a file, *Widell* and *Lordi* cannot be combined as suggested by the Examiner, and the combination of *Widell*, *Lordi*, and *Carter* does not teach "change a first set of data for a first thread associated with a first file of said filesystem" in combination with wherein the first file comprises an inode page, and wherein the second file comprises a directory page. Therefore, the combination does not render Applicants' claims 6, 7, 14 and 15 obvious.

II. Conclusion

It is respectfully urged that the subject application is patentable over *Widell, Lordi* and *Carter* and is in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: February 27, 2007

Respectfully submitted,

/Lisa L.B. Yociss/
Lisa L. B. Yociss
Reg. No. 36,975
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants